## SPECIFICATION AMENDMENTS

Please amend the paragraph beginning on page 10, line 8, as follows:

Fig. 6 illustrates a generalized method that can be employed in accordance with the present invention to maintain data integrity while updating a shared data element group such as the cyclic graph 20. In an initialization step 30, a global generation number is established relative to the data element group and each data element in the group is assigned a copy of the global generation number at the time of its creation. In steps 32 and 34, an updater that wishes to replace, delete or insert a group data element generates a new data element and sets its generation number field to a value that is one greater than the current global generation number. If the updater is replacing a current data element, the new data element will be a modified copy of the current data element. If the updater is deleting a current data element, the new data element will be a copy of the current data element with a "deleted" flag set. If the updater is inserting a new data element, the new data element is created from scratch. In step 36, the updater sets version links between the new data element and its pre-update version (if such a version exists). In particular, each data element version maintains a set of two version pointers, one being an old-version pointer to a previous version of the data element (if any) and the other being a new-version pointer to a next version of the data element (if any). A NULL old-version pointer is used for any data element having no previous version (i.e., the data element has not been updated since the last grace period or is an insert). A NULL new-version pointer is used for any data element having no next version (i.e., it is the most current version). In step 38, the updater changes any link pointers that point to the old version of the data element to instead point to the new version, and then increments the global generation number.

Please amend the paragraph beginning on page 20, line 3, as follows:

Turning now to Figs. 14A- 14E, the use of pointer forwarding entities is presented in the context of a circular linked list 160. The linked list comprises three data elements A, B and C in that order. There is also a global list head pointer forwarding entity P(h) that points to data element A, and three additional pointer forwarding entities P(A), P(B) and P(C) respectively pointing ~~to~~from data elements A, B and C to data elements B, C and A. It is assumed that data elements A and B are to be interchanged in the list 160. One way to make this change would be to allocate new copies of A, B, and C, then update their pointers and version number, proceeding as one would for a normal atomic replacement of these three elements. However, it is sometimes desirable to switch the two elements without creating new copies. Pointer-forwarding entities allow the updater to accomplish the goal, and the present example thus illustrates a further benefit of the pointer-forwarding approach.